Enumeration of Circuits and Minimal Forbidden Sets

Enumeration of Circuits and Minimal Forbidden Sets

Frederik Stork [a] Marc Uetz [b]

[a] *ILOG Deutschland GmbH, Germany* (fstork@ilog.de)

[b] *Universiteit Maastricht, The Netherlands* (m.uetz@ke.unimaas.nl)

Abstract

In resource-constrained scheduling, it is sometimes important to know all inclusion-minimal subsets of jobs that must not be scheduled simultaneously. These so-called minimal forbidden sets are given implicitly by a linear inequality system, and can be interpreted more generally as the circuits of a particular independence system. We present several complexity results related to computation, enumeration, and counting of the circuits of an independence system. On this account, we also propose a backtracking algorithm that enumerates all minimal forbidden sets for resource constrained scheduling problems.

Key words: Independence system, Circuit, Enumeration, Minimal forbidden set

1 Introduction

Given a finite ground set V, an independence system is defined as a family $\mathcal{I}$ of subsets of V with two properties. First, $\emptyset \in \mathcal{I}$, and second, any subset of any member of $\mathcal{I}$ also belongs to $\mathcal{I}$. The sets in $\mathcal{I}$ are called independent sets, and the inclusion-maximal independent sets are the *bases* $\mathcal{B}$ of $\mathcal{I}$. The sets not in $\mathcal{I}$ are called dependent sets, and inclusion-minimal dependent sets are the *circuits* $\mathcal{C}$ of $\mathcal{I}$; see also [6]. Given a membership oracle for such an independence system $\mathcal{I}$, we are interested in the problem to enumerate all circuits of $\mathcal{I}$. It is obvious that the output size may be exponential in terms of the size of V. The complexity is thus measured in terms of the size of both, in– and output. Given that $|V| = n$, and given a (poly(n)) membership oracle for some collection of subsets $\mathcal{S} \subseteq 2^V$, we use the following definitions; see [4].

Definition 1 *The enumeration problem for $\mathcal{S}$ is solvable in* polynomial total time *if there exists an algorithm and a polynomial $p(\,\cdot\,,\,\cdot\,)$, such that the algorithm correctly outputs all members of $\mathcal{S}$ in $p(n, |\mathcal{S}|)$ time.*

Given a sub-collection $\mathcal{X} \subseteq \mathcal{S}$, the *increments problem* is the problem: either decide that $\mathcal{X} = \mathcal{S}$, otherwise output a new element in $\mathcal{S} \setminus \mathcal{X}$.

Definition 2 *The enumeration problem for $\mathcal{S}$ is solvable in* incremental polynomial time *if there exists an algorithm and a polynomial $p(\cdot\,,\cdot)$, such that the algorithm correctly solves the increments problem in $p(n, |\mathcal{X}|)$ time, for any $\mathcal{X} \subseteq \mathcal{S}$.*

If the enumeration problem for $\mathcal{S}$ is solvable in incremental polynomial time, it is also solvable in polynomial total time: Starting with $\mathcal{X} = \emptyset$, the iterative solution of the increments problem yields the complete collection $\mathcal{S}$, in time polynomial in n and $|\mathcal{S}|$. The reverse, however, need not be true.

2 The general case

The following theorem is well known; it is proved using a reduction from the NP-complete decision problem SATISFIABILITY [3].

Theorem 3 ([5]) *Unless P=NP, there does not exist a polynomial total time algorithm that enumerates the bases $\mathcal{B}$ of any independence system $\mathcal{I}$.*

Likewise, using a simple duality argument, we can show the following.

Theorem 4 *Unless P=NP, there does not exist a polynomial total time algorithm that enumerates the circuits $\mathcal{C}$ of any independence system $\mathcal{I}$.*

The proof uses the fact that the circuits of $\mathcal{I}$ are the bases of the following, dual independence system: $\mathcal{I}^D = \{W \subseteq V \mid V \setminus W \notin \mathcal{I}\}$. These two theorems say that, unless P=NP, an algorithm cannot exist which solves the problem for *any* independence system $\mathcal{I}$. For particular realizations of the membership oracle of $\mathcal{I}$ however, efficient enumeration algorithms may well exist.

3 Scheduling and linear inequality systems

In resource-constrained scheduling, the input consists of a set of partially ordered jobs $(V, \prec)$ (the partial order representing the precedence constraints) and resource constraints. The latter are given through a number of resource types k with availabilities b_k, and resource requirements a_{kj} of these resource types for all jobs $j \in V$. If a subset S of jobs consumes more of a resource type than available, the respective jobs in S may not be processed in parallel. The subsets of jobs that may be processed in parallel define an independence system. The circuits of this independence system are either pairs of (precedence-)related jobs, $\{i, j\}$ with $i \prec j$, or the so-called *minimal forbidden sets* (minimal anti-chains of $(V, \prec)$ which may not be processed in parallel). For several algorithmic purposes, e.g. in stochastic scheduling [8], a complete list of the minimal forbidden sets is required. This amounts to the computation of the circuits of the corresponding independence system. The membership oracle of this system is a linear inequality system $Ax \leq b$, where A, b contains one row in for each resource type k, and one row for each edge in the comparability graph of the partial order $(V, \prec)$. The circuits are the minimally infeasible $\{0,1\}$–vectors for $Ax \leq b$; they are the incidence vectors of either pairs of (precedence-)related jobs, or minimal forbidden sets.

Inspired by [2], and using a reduction from the NP-complete decision problem Independent Set in graphs [3], we show:

Theorem 5 *Unless P=NP, there does not exist a polynomial total time algorithm that enumerates the minimally infeasible* $\{0,1\}$*–vectors of an arbitrary linear inequality system* $Ax \leq b$*.*

In fact, in [2] it is shown that the decision version of the corresponding increments problem is NP-complete. What is interesting is the fact that the 'dual' problem is apparently much easier: It follows from [2] that the increments problem for the maximally feasible $\{0,1\}$–vectors of an arbitrary linear inequality system $Ax \leq b$ can be solved in quasi-polynomial time, hence there is also a quasi-polynomial total time algorithm for this problem. (Such a result is not likely for the problem considered here, because then all NP-hard problems could be solved in quasi-polynomial time.)

In terms of scheduling, Theorem 5 immediately yields:

Corollary 6 *Unless P=NP, there does not exist a polynomial total time algorithm that enumerates the minimal forbidden sets for any instance of the resource-constrained project scheduling problem.*

It turns out that even the computation of the *number* of minimal forbidden sets is a hard problem, because we can show:

Theorem 7 *The problem to compute the number of minimally infeasible* (*maximally feasible*) $\{0,1\}$*–vectors of an arbitrary linear inequality system* $Ax \leq b$ *is* #P*-complete.*

The proof uses a reduction from the problem to compute a maximum cardinality anti-chain of a partial order. While this problem is polynomially solvable, the associated counting problem is known to be #P-complete [7].

Nevertheless, for practical purposes we implemented a simple backtracking algorithm that lists the minimal forbidden sets for any instance of the resource-constrained project scheduling problem. In general, this algorithm can have an exponential running time in terms of in– and output of the problem. Yet, empirically it improves considerably upon a divide-and-conquer algorithm previously suggested in [5,1]; see [9]. Moreover, we can show that the algorithm is efficient for an important special case.

Proposition 8 *There exists an incremental polynomial time* (*hence also a polynomial total time algorithm*) *that enumerates the minimal forbidden sets for any instance of the resource-constrained project scheduling problem, given that the number of resource types is* 1.

We refer to [9] for several other results related to enumeration and computation of minimal forbidden sets, as well as detailed computational results.

Acknowledgements The authors thank Marc Pfetsch for many insightful discussions on the topic, which eventually lead to Theorem 5.

References

[1] M. Bartusch, Optimierung von Netzplänen mit anordnungsbeziehungen bei knappen Betriebsmitteln, Ph.D. thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, Germany (1984).

[2] E. Boros, K. Elbassioni, L. Khachiyan, K. Makino, Dual-bounded generating problems: All minimal integer solutions for a monotone system of linear inequalities, SIAM Journal on Computing 31 (2002) 1624–1643.

[3] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.

[4] D. S. Johnson, M. Yannakakis, C. H. Papadimitriou, On generating all maximal independent sets, Information Processing Letters 27 (1988) 119–123.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Generating all maximal independent sets: NP-hardness and polynomial-time algorithms, SIAM Journal on Computing 9 (1980) 558–565.

[6] J. G. Oxley, Matroid Theory, Oxford University Press, 1992.

[7] J. S. Provan, M. O. Ball, The complexity of counting cuts and of the probability that a graph is connected, SIAM Journal on Computing 12 (1983) 777–788.

[8] F. Stork, Stochastic resource-constrained project scheduling, Ph.D. thesis, Institut für Mathematik, Technische Universität Berlin, Berlin, Germany (2001).

[9] F. Stork, M. Uetz, On the enumeration of minimal covers and minimal forbidden sets, METEOR Research Memorandum 02/036, Universiteit Maastricht, Maastricht, The Netherlands (2002).